



Módulos e Matrizes em Python

Bem-vindo ao nosso estudo sobre a Módulos e Matrizes em Python. Este estudo ajudará você a desenvolver programas na linguagem Python com as principais estruturas de controle e modularizados, dando uma boa base para você criar programas de Inteligência Artificial, mais à frente. Bons estudos!

Array

Um array é uma coleção de elementos do mesmo tipo. Por isso chamamos de variável homogênea. O Python até permite mais de um tipo de dado num array, mas é mais comum utilizar dados do mesmo tipo. Assim como a lista, um array é mutável, isto é, os seus elementos podem ser alterados.

Vetores e Matrizes em Python

Uma matriz é um conjunto de dados dispostos em linhas e colunas. Um vetor é uma matriz que possui uma única linha com várias colunas. Dissemos que o vetor é uma matriz unidimensional, isto é, de apenas uma dimensão. A matriz pode ter quantas dimensões for necessário, mas o nosso estudo está focado em matrizes unidimensionais e bidimensionais, apenas.
Segue abaixo um exemplo de matriz unidimensional (vetor).



```
# Exemplo de vetor
num = [1, 2, 3, 4, 5]
print (num)
#Mostrando pelo for:
for x in num:
    print(x,end=' ')
```

Esse código cria um vetor de números, inicializando com 5 números. Os números são mostrados de duas formas diferentes. Podemos utilizar um for específico para coleções, como fazemos com listas. No print foi utilizado o parâmetro end para que os números sejam apresentados na mesma linha. O padrão do comando print é mostrar um em cada linha, mas como o atributo end=' ', estamos substituindo a quebra de linha pelo espaço. Veja o resultado:

```
[1, 2, 3, 4, 5]
1 2 3 4 5
```

Podemos acessar cada elemento do vetor através de seu índice, por exemplo em num[0] está armazenado o número 1, em num[4] está armazenado o valor 5.

Para receber dados do usuário, armazenar num vetor e acessar este vetor para apresentar os dados para o usuário, em Python, podemos realiz utilizando uma estrutura de repetição da seguinte forma:

```
num = [0,0,0,0,0]
for i in range(5):
    num[i] = int(input('Digite um número: '))

print(num)
```

Resultado do processamento:

```
Digite um número: 10
Digite um número: 20
Digite um número: 30
Digite um número: 40
Digite um número: 50
[10, 20, 30, 40, 50]
```

Vamos entender agora sobre uma matriz bidimensional, que tem linhas e colunas.

Em Python, para declarar, atribuir elementos e mostrar estes dados de uma matriz, podemos realizar da seguinte forma:

```
# Exemplo de Matriz:
matriz = [[0,0],[0,0],[0,0]]
print(matriz)
for i in range(3):
    for j in range(2):
        matriz[i][j] = int(input('Digite um número: '))
print(matriz)
```

Para acessar um elemento específico da matriz, utilizamos da seguinte forma:
print(matriz[1][1])
Esse comando vai mostrar o elemento que está na linha de índice 1, coluna de índice 1, portanto mostrará o número 4.

Modularização

Quando desenvolvemos um programa Python, precisamos nos preocupar em como desenvolvê-los de forma que fique simples de ser entendido. Para isso, podemos, ao invés de criar um único programa principal, quebrá-lo em partes e simplificar o problema, para depois, quando reunidos, resolverem o problema maior.
Para que fique menos complexo o programa, podemos dividi-lo em pequenos módulos, que resolvem problemas menores. Podemos escrever quatro tipos de módulos:
Procedimento sem parâmetro: um módulo que não recebe nada de parâmetro e não retorna nada. Exemplo:

```
def ola():
    print("Olá, bem-vindo ao estudo de modularização em Python")

# chamada do módulo:
ola()
```

Neste exemplo, o módulo ola() recebe o nome como parâmetro para utilizar na mensagem. E no módulo principal, o nome é digitado pelo usuário e enviado para o módulo ola().

Resultado de um processamento:



```
Digite o seu nome: Maria de Souza
Olá, Maria de Souza seja bem-vindo(a)!
Fim do processamento
```

Função sem parâmetro: É um módulo que não recebe parâmetros, mas retorna um valor. Exemplo:

```
def soma():
    n1 = int(input('Digite um número: '))
    n2 = int(input('Digite outro número: '))
    s = n1+n2
    return s

#módulo principal:
s = soma()
print('Resultado da soma: ',s)
```

Neste exemplo, a função soma() não recebe nada como parâmetro, mas em seu interior faz a entrada de dois números, soma-os e retorna o seu resultado. Veja que na chamada da função, o resultado será armazenado na variável s. Veja o resultado do processamento:

```
Digite um número: 10
Digite outro número: 20
Resultado da soma:  30
```

Função com parâmetro: É um módulo que recebe parâmetro e retorna o seu resultado. Veja o exemplo:

```
def soma(n1, n2):
    s = n1+n2
    return s

#módulo principal:
n1 = int(input('Digite um número: '))
n2 = int(input('Digite outro número: '))
s = soma(n1, n2)
print('Resultado da soma: ',s)
```

Neste exemplo, a função soma() recebe os números como parâmetro, soma-os e retorna o resultado. Veja o resultado do processamento:

```
Digite um número: 30
Digite outro número: 50
Resultado da soma:  80
```

Como vimos, podemos simplificar um programa complexo, dividindo-o em pequenos módulos mais simples, para chegar ao resultado com a jun desses módulos. Durante a jornada profissional de um programador, a criação de módulos será parte importante de sua rotina de trabalho.

Atividade Extra

Para se aprofundar no assunto desta aula, pesquise o que é uma função recursiva e pratique a modularização com vários exemplos diferentes dos mostrados nessa aula.

Referência Bibliográfica

MENEZES, Nilo N. C. **Introdução à Programação com Python:** Algoritmos e lógica de programação para iniciantes. São Paulo: Novatec, 2014.

SUMMERFIELD, Mark. **Programação em Python 3:** uma introdução completa à linguagem Python. Rio de Janeiro: Alta Books, 2012.

Ir para exercício